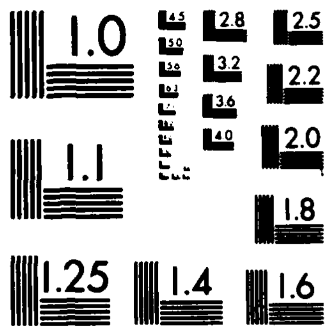
AD-A184 643 A DIGIT PIPELINED DYNAMIC TIME WARP PROCESSOR(U) 1/1
PENNSYLVANIA STATE UNIV UNIVERSITY PARK DEPT OF
COMPUTER SCIENCE M J IRWIN AUG 86 CS-86-23
UNCLASSIFIED ARO-20090 23-EL DAAG29-83-K-0126 F/G 12/6 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DTIC FILE COPY

UNCLASSIFIED

AD-A184 643

DTIC ELECTE SEP 1 4 1987 S D

REPORT DOCUMENTATION PAGE

1. REPORT NUMBER ARO 20090.23-EL	2. GOVT ACCESSION NO. N/A	3. RECIPIENT'S CATALOG NUMBER N/A
4. TITLE (and Subtitle) A DIGIT PIPELINED DYNAMIC TIME WARP PROCESSOR		5. TYPE OF REPORT & PERIOD COVERED Internal Technical
		6. PERFORMING ORG. REPORT NUMBER N/A
AUTHOR(s) M.J. Irwin		8. CONTRACT/GRANT NUMBER(s) DAAG29-83-K-0126
PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science The Pennsylvania State University University Park, PA 16802		10. PROGRAM WORK UNIT NUMBERS N/A
CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE August 1986
		13. NUMBER OF PAGES - 24
MONITORING AGENCY NAME & ADDRESS N/A		15. SECURITY CLASS. (of this report) Unclassified
DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the Abstract entered in Block 20) N/A		
18. SUPPLEMENTARY NOTES The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.		
19. KEY WORDS Isolated word recognition, dynamic time warping, digit pipelining, signed-digit representation, most significant digit first processing, VLSI		

20. ABSTRACT

A custom CMOS VLSI processor is presented which can achieve real-time isolated word recognition for dictionaries of up to 2000 words. The processor is based on the dynamic time warping (DTW) algorithm, an exhaustive search technique which permits nonlinear pattern matching between an unknown utterance and a reference word. Our design differs from previous DTW designs in that (1) all data is represented in signed-digit, base 4 format; (2) digits are passed between processing elements in a most significant digit first, digit serial fashion; and (3) the algorithms are pipelined at the digit level. Because of these features, nine processing elements will fit on one chip using 3 micron feature size devices and an 84 pin package. The VLSI DTW processor presented is both flexible and modular. The design is independent of the number of coefficients per frame and the precision of those coefficients. The design is also easily expandable in the number of frames per word and the warp factor used to achieve the nonlinear matching.

UNCLASSIFIED

ARO 20090.23-EC

A Digit Pipelined Dynamic Time Warp Processor

Mary Jane Irwin

CS-86-23 August 1986

Department of Computer Science
The Pennsylvania State University
University Park, PA 16802

A DIGIT PIPELINED DYNAMIC TIME WARP PROCESSOR

Mary Jane Irwin
Department of Computer Science
The Pennsylvania State University
University Park, PA 16801

Abstract

A custom CMOS VLSI processor is presented which can achieve real-time isolated word recognition for dictionaries of up to 2000 words. The processor is based on the dynamic time warping (DTW) algorithm, an exhaustive search technique which permits nonlinear pattern matching between an unknown utterance and a reference word. Our design differs from previous DTW designs in that (1) all data is represented in signed-digit, base 4 format; (2) digits are passed between processing elements in a most significant digit first, digit serial fashion; and (3) the algorithms are pipelined at the digit level. Because of these features, nine processing elements will fit on one chip using 3 micron feature size devices and an 84 pin package. The VLSI DTW processor presented is both flexible and modular. The design is independent of the number of coefficients per frame and the precision of those coefficients. The design is also easily expandable in the number of frames per word and the warp factor used to achieve the nonlinear matching.

Index Terms

Isolated word recognition, dynamic time warping, digit pipelining, signed-digit representation, most significant digit first processing, VLSI

Introduction

A key component of any speech recognition system is the hardware which matches an unknown utterance in a speech sample against a dictionary of words. The input speech waveform must first be filtered, correlated, analyzed, partitioned into distinct utterances, and time normalized before this pattern matching can begin. Isolated word recognition can be performed, for example, by using beam searching [AnB], two-level DP matching [Sak], or dynamic time warping (DTW) [e.g., Ack, SaC, YoJ]. These algorithms differ in their search techniques (e.g. exhaustive or best first) and in their scoring methods (e.g., best match or maximum likelihood). However it is accomplished, isolated word recognition is one of the most time consuming of all the tasks in serial word recognition systems. Of course, real-time recognition is a goal. A custom VLSI design for a DTW processor is proposed which is real time for dictionaries containing up to 2000 words.

A block diagram of that portion of the speech recognition system we are concerned with in this paper, isolated word recognition, is shown in Figure 1. The parameters characterizing the unknown utterance have been time normalized into a set of I frames ordered in time. A time normalization processor has been used to stretch or shrink the unknown utterance to this fixed length. In our DTW processor we assume utterances have been time normalized into 42 frames [YoJ], although our design is modular and extendable. By varying the number of VLSI packages, a DTW processor with any multiple-of-three number of frames can be assembled. The parameters of each frame are the coefficients characterizing the unknown utterance at that point in time. Linear predictive coding [RaS] can be used to produce these frame coefficients, typically between four and sixteen coefficients per frame. Each of these frame coefficients is a 6-bit to 24-bit positive (or unsigned) integer. We assume 16-bit coefficients for our initial timing estimates. However, the hardware in the DTW processor has been designed in such a way as to be independent of both the number of coefficients per frame and the number of bits used to represent each coefficient. Only the recognition rate is affected. The reference dictionary stores the coefficients of the words against which the utterance is to be matched. Each of these reference words has also been time normalized to I frames of four to sixteen, 6-bit to 24-bit positive integer coefficients. We assume a reference dictionary of up to 2000 words. (Webster's Ninth New Collegiate Dictionary contains over 160,000 words.)

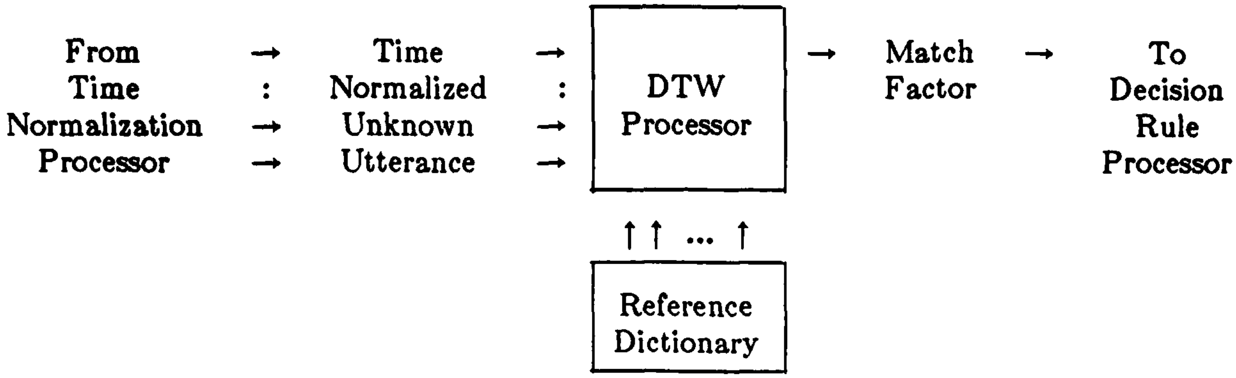


Figure 1. Isolated Word Recognition

Our goal was to design an isolated word recognition VLSI processor which could achieve real-time processing for reference dictionaries of up to 2000 words. The input sampling rate typically ranges from 6.25 KHz to 20 KHz [YoJ]. After this input speech waveform has been processed and time normalized into distinct utterances, words are supplied to the isolated word recognition processor (the DTW processor in Figure 1) at the average rate of one utterance every half second [YoJ]. As the unknown utterance is matched against an entry in the reference dictionary a match factor is generated. This match factor consists of between four and sixteen, 6-bit to 24-bit positive integers indicating how close each of the characterizing coefficients matched, "summed" across the I frames. A match factor is produced for each word in the reference dictionary. A decision rule processor then uses these match factors to select the closest match. A match of an unknown utterance against the entire dictionary contents must be done in half a second or less to meet the real-time goal.

Dynamic Time Warp Processor

Our isolated word recognition processor is based on a DTW algorithm [Ack, BAW, KNL, MaR, Owe, QGG, SaC, WBA, YoJ], an exhaustive search technique which permits nonlinear pattern matching between the unknown utterance and the words in the dictionary by allowing for natural timing variations between the unknown utterance and the reference words. An illustration of a simple DTW processor for a four frame utterance is shown in Figure 2. The horizontal axis represents the reference words output from the dictionary which have been normalized into four frames (R_0 through R_3) each representing a successive time interval. Associated with each frame are the coefficients characterizing the word at that point in time. The vertical axis represents the unknown utterance which has been time normalized into four frames (U_0 through U_3) of coefficients. This simple DTW processor contains a grid of sixteen processing ele-

ments numbered PE(0,0) through PE(3,3). Each PE is connected to six of its nearest neighbors.

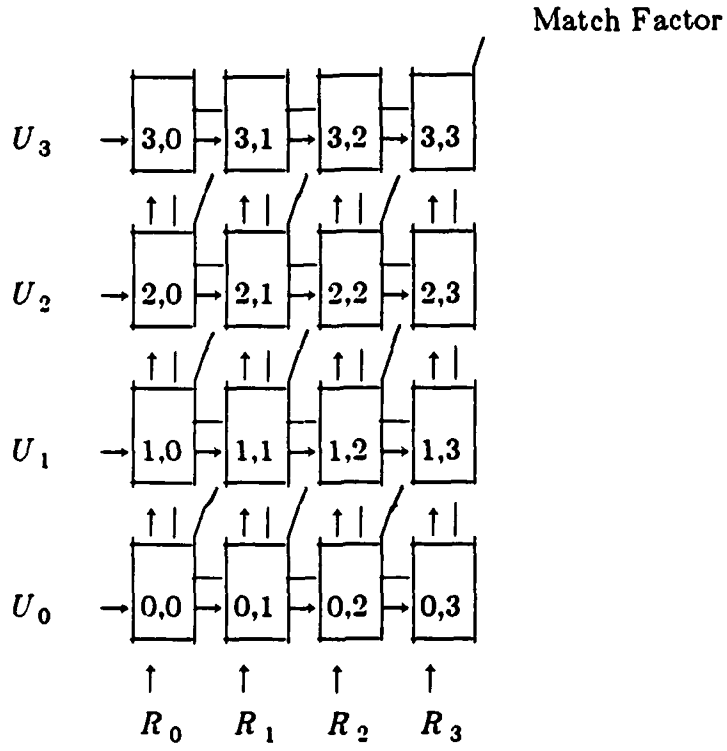


Figure 2. A Simple DTW Processor (I = 4)

A single PE from the simple DTW processor of Figure 2 is shown in Figure 3. Each PE must be able to compute

$$S_{i,j} = D_{i,j} + \text{minimum}(S_{i-1,j}, S_{i-1,j-1}, S_{i,j-1})$$

where $D_{i,j}$ is the local "distance" between a coefficient of frame j of the reference word and a coefficient of frame i of the unknown utterance and $S_{i,j}$ is the accumulated minimum "distance" along a path of adjacent PE's. We use the simple local distance measure of the absolute value of the difference of R_j and U_i. Other choices for computing the local distance are to use a squared Euclidean distance measure [MaR] or a logarithmically scaled distance measure [YoJ]. Weighting factors, usually some multiple of two, can also be applied to the local distance measures [MaR, MRR, RaS]. We use a weighting factor of one only. In Figures 2 and 3, the computed values which are passed between neighboring PE's are indicated by — , /, and | lines, while the coefficient which are merely passed through a PE to its neighbor are indicated by → and ↑ arrows.

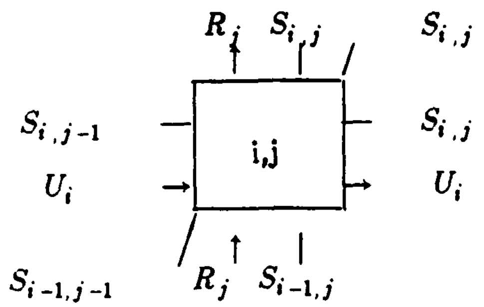


Figure 3. One DTW Processing Element

Dynamic programming [Bel] is used to find the minimum distance path from PE (0,0) to PE (I-1,I-1). By imposing a slope constraint on this path, only nearest neighbor interconnect of PEs is necessary [MaR]. Fortunately, since speech is essentially "well behaved," limiting the slope of the path does not adversely affect the recognition accuracy. (If weighting factors are applied to each branch of the path, a bias in the path movement through the mesh can also be imposed.) Parallel computation proceeds in systolic waves from lower left to upper right in the grid of PE's in Figure 2, where all PE's on the same back diagonal simultaneously execute in the same wave. Four to sixteen accumulated minimum distances (one for each coefficient) which are the sum of the minimum local distances along a path from PE (0,0) to PE (I-1,I-1) constitute the match factor which tells how close the the match was between the two words. In this way, a match factor is produced for every word in the dictionary.

In the DTW processor shown in Figure 2, there may be no need to match R_3 against U_0 if the time discontinuity between the reference word and the unknown utterance can never be that large. In this case, PE (3,0) and PE (0,3) are unnecessary. A "warping factor" of two results; the reference frame coefficients and the unknown frame coefficients two before and two after the intersecting PE may be included in the accumulate minimum distance. With a warp factor of one, PE's (3,0), (3,1), (2,0), (0,3), (1,3) and (0,2) would be unnecessary.

We assume a warp factor of six [YoJ] in our DTW processor, although once again by varying the amount of hardware any multiple-of-three warp factor could be accommodated. The entire DTW processor for an I of 42, a warp factor of six, a variable number of coefficients per frame, and a variable number of bits per coefficient is depicted in Figure 4. This DTW processor design uses 504 PEs.

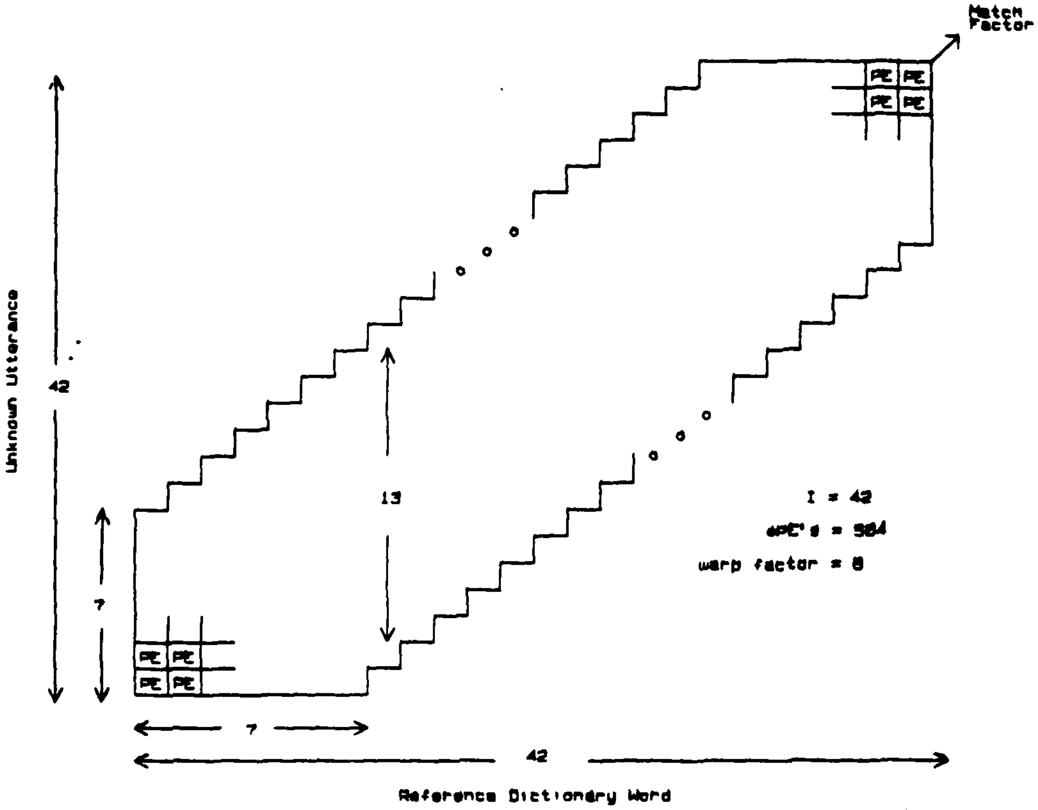


Figure 4. The DTW Processor

Several DTW processors for isolated and connected word recognition (and other pattern matching problems) have been proposed and implemented in VLSI [Ack, BAW, KNL, MaR, Owe, QGG, WBA]. In the AT&T DTW processor [Ack, BAW, WBA] as in many of the other VLSI DTW designs, one overriding consideration was the need to integrate at least one, and preferably more than one, PE onto a single VLSI chip. With one PE per chip, a minimum of four data ports are required (the diagonal interconnect can be omitted by routing $S_{i-1,j-1}$ through a north or east neighboring PE). Because of this large I/O requirement for a 64 pin package, a bit serial communication scheme was used. The 16-bit coefficient operands are streamed into the chip in a bit serial, least significant bit first fashion. Although the local distance could be computed as the bits are input using a conventional serial adder/subtractor, to do the minimum operation as the bits are being input requires the operands to be delivered most significant bit first. Apparently, these two primitive operations have conflicting I/O requirement, one requiring its operands least significant bit first and one requiring them most significant bit first. Of course, both operations can be done if the full precision operand values are available. So, in the AT&T DTW processor the operands are first collected into 16-bit

input registers, then the distances are computed using the full precision values. After processing, the operands are streamed out of the chip once again in a bit serial, least significant bit first f·shion. While this drastically reduces the interconnect, it also limits performance since it takes 16 clock cycles to transfer one operand.

Our VLSI DTW processor differs from previous designs in that the data is passed between PE's as base 4, signed-digits [Atk], most significant digit first. Base 4, serial signed-digit arithmetic components [IrO] are used to compute the distance digits as the operand digits are input. For each pair of operand digits input (except the first, most significant digit) an output digit is produced. As soon as an output digit has been generated by one PE, it is passed along to the next PE as an input. By pipelining operations at the digit level [IrO] in this way, the speed limitations of serial communication is obviated. A very high degree of concurrency and a high data rate can be maintained, while the pin out requirements are kept low.

In the following section, the functioning of the digit serial primitive arithmetic components required in a DTW processor PE are described. A PE is characterized by its I/O requirements and size (dimensions in λ for CMOS custom layout). The full DTW processor is then considered with an appropriate partitioning into VLSI chips using MOSIS standard frame sizes [MOS]. Finally some speed estimates for the isolated word recognition problem are given.

Digit Serial Primitive Arithmetic Components

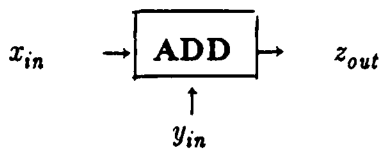
In our DTW processor the data is transferred between PE's digit serially, where a digit is a base 4 signed-digit. (In our notation we use capital letters to denote full precision operands and small letters to denote digit values.) Computed values are produced in the maximally redundant symmetric signed-digit set for base 4, the digit set {-3, -2, -1, 0, 1, 2, 3}. For example, in this digit set the numerical value 63_{10} can be represented in four digits as 0333_4, $1\bar{1}33_4$, $10\bar{1}3_4$, or $100\bar{1}_4$ where $\bar{1} = -1$. Input values, the dictionary and unknown utterance coefficients, are represented in the conventional base 4 digit set, the digit set {0, 1, 2, 3}. Since this is a subset of the signed-digit set used in the DTW processor, no conversion on input is required. The match value is produced in signed-digit representation. Two's complement encoding is used to encode the digits into a binary form. Thus, three wires are required to transmit a distance digit, while only two wires are required to transmit a coefficient digit. For example, $10\bar{1}3_4$ would be encoded in 12 bits as $001\ 000\ 111\ 011_2$. The encoding "100" is not used and thus is available for signaling overflow, NAN (not a number) conditions, or for use in hardware error detection. All data has fixed and constant word length of p base 4 di-

gits; for the speech recognition problem $p = 8$ for 16-bit coefficient, although we will use $p = 4$ in the numerical examples in this section for simplicity. Thus, 24-bits are required to encode a conventional 16-bit value into a signed-digit representation. In signed-digit representation the sign of the operand is the sign of the most significant digit; i.e., a separate sign digit is not required. Also, there is only one representation for the operand zero - the digit string consisting of all zero.

The data flow in the DTW processor is digit serial with the most significant digit being transmitted first. The primitive arithmetic components with this data flow characteristic required to build the DTW PE are a digit serial add/subtract component, a digit serial minimum component, and a digit serial absolute value component. Operand digits are input to a primitive arithmetic component one set per clock cycle. The most significant result digit is generated one clock cycle after the most significant digits of the operand(s) are input. After the most significant result digit is output, a result digit is output for each subsequent cycle. This time between when the first digit of data is input and when the first digit of the output is generated is the component latency. In our design all primitive components have a fixed latency of one. Furthermore, the operations are pipelined at the digit level. When a digit of result is output from one primitive arithmetic component it is passed immediately to the next as an input.

Add/Subtract Component

The digit serial add/subtract component is defined in Figure 5. A generic add/subtract component is discussed here, but in the DTW processor separate add and subtract components are needed. In Figure 5, the operator "o" represents either a subtraction operation for computing the local distance or an addition operation for forming the accumulated minimum distance. When computing the local distance, $x_{in} = u_{in}$, and $y_{in} = r_{in}$ which are both in the conventional digit set {0, 1, 2, 3}. When computing the accumulated minimum distance, x_{in} is the digit value computed by the local distance absolute value subtraction operation and y_{in} is the digit value computed by the digit serial minimum operation, both of which are in the digit set {-3, -2, -1, 0, 1, 2, 3}.



where

(1) $s_j = x_{in} \ o \ y_{in} - 4\, c_j$

where $-1 \leq c_j \leq 1$ is chosen so that $-2 \leq s_j \leq 2$

(2) $z_{out} = s_{j-1} + c_j$

Figure 5. MSD Digit Serial Adder/Subtractor

The present interim sum digit, s_j, which is determined by the first digit operation (1), is stored in an internal latch to be used in the next cycle. Then a second digit operation (2) adds the present interim carry, c_j, to the previous interim sum and this result is output. Most importantly this addition is *guaranteed* to be carry free because of the restricted digit sets allowed for the interim carry and the interim sum. The latency of the adder/subtractor primitive component is one; one clock cycle after the most significant digits of the operands are input the most significant digit of the result is output.

Table 1 gives the input digits/interim digits mapping for the first digit operation, (1) in Figure 5. The local distance computation, $u_{in} - r_{in}$, uses only the subset of the table from -3 to 3.

Table 1. **ADD** Operation (1) Mapping

x_{in} o y_{in}	c_j		s_j
-6	-1		-2
-5	-1		-1
-4	-1		0
-3	-1		1
-2	0		-2
		or	
-2	-1		2
-1	0		-1
0	0		0
1	0		1
2	0		2
		or	
2	1		-2
3	1		-1
4	1		0
5	1		1
6	1		2

The second digit addition mapping is straightforward. The addition component requires two digit adder cells and a one digit storage cell as shown in the block diagram of Figure 6. (The first digit adder cell is decomposed further into a digit adder sum cell (1s) and a digit adder carry cell (1c).) Each of these cells can be built in four levels of two input logic gates.

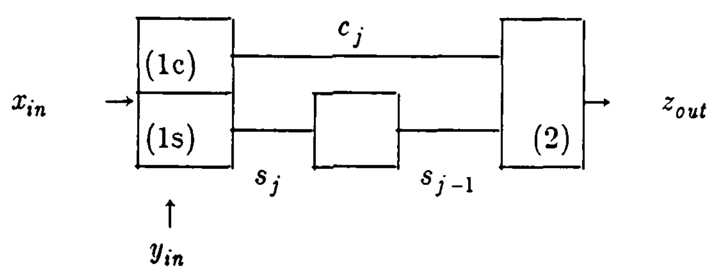


Figure 6. **ADD** Logic Block Diagram

Cells of the CMOS adder component have been fabricated by MOSIS using 3 micron feature size, double layer metal CMOS mesh arrays [Bee]. Preliminary tests on these fabricated chips lead us to believe that a gate delay of less than two nsec can be achieved. The mesh array layout plot of the adder primitive component is shown in Figure 7. It is approximately 200λ by 200λ in size. Only *one* primitive adder component is needed regardless of the operand precision.

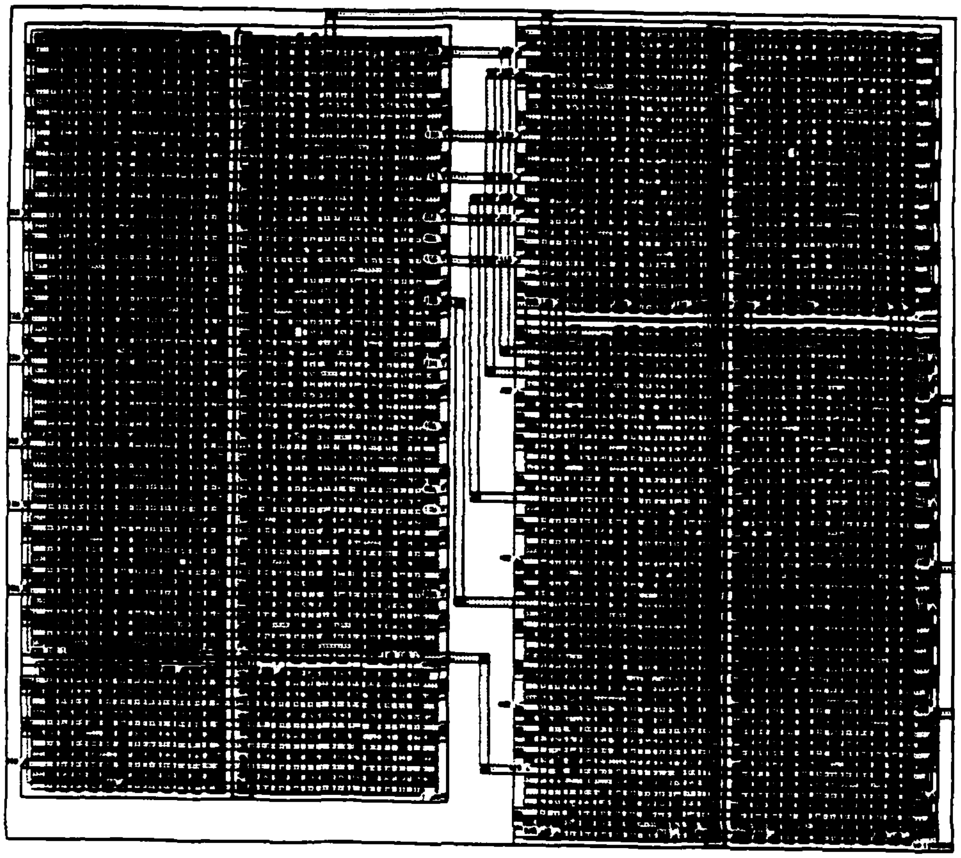


Figure 7. Layout of **ADD**

Examples of msd, base 4, digit serial addition are given in Table 2 for $X_{in} = 1223_4$ and $Y_{in} = 1213_4$ followed immediately by $X_{in} = 12\bar{3}2_4$ ($= 1112_4$) and $Y_{in} = 112\bar{1}_4$ ($= 1113_4$). The correct first sum $Z_{out} = 3102_4$ which is produced by the adder would incur a full precision carry if performed on a ripple-carry adder. The second sum is correctly formed as $Z_{out} = 3\bar{1}\bar{1}1_4$ ($= 2231_4$). An "*" in the example indicates that that value is not used in the computation. For a sequence of additions (or subtractions), the msd's of the operands for the next addition can be input at the next clock after the lsd's of the operands for the present addition have been input. As long as overflow is not allowed, the interim carry formed during operation (1) for the next addition is guaranteed to be zero. Note that this interim carry is the one used in operation (2) to complete the present addition and if it was not zero an incorrect lsd for the present addition would be output.

Table 2. Digit Serial Addition Example

clock	1	2	3	4	5	6	7	8	9
x_{in}	1	2	2	3	1	2	-3	2	*
y_{in}	1	2	1	3	1	1	2	-1	*
c_j	0	1	1	1	0	1	0	0	0
s_j	2	0	-1	2	2	-1	-1	1	*
z_{out}	*	3	1	0	2	3	-1	-1	1

The primitive arithmetic components of the DTW processor have been designed assuming that intermediate results will never overflow. If overflow does occur, the least significant digit of the previous intermediate result is contaminated. This situation can be avoided by having the **ADD** component which is used to compute the accumulated distance measure "corrupt" the most significant *result* digit *on* overflow. Since the DTW processor is looking for the closest match (i.e., the smallest match factor), this result will not be used anyway, so an exact result is not required. Overflow can be inhibited altogether by using the digit mapping shown in Table 3 for digit operation (1) in the **ADD** component, but *only* for the most significant digit. As we will see, the arrival of the most significant digit of an operand coincides with a wave control signal, W_{in}, one of which is broadcast along each back diagonal in the PE array.

Table 3. Corrupted MSD **ADD** Operation (1) Mapping

x_{in} o y_{in}	c_j	s_j
-6	0	-2
-5	0	-2
-4	0	-2
-3	0	-2
-2	0	-2
-1	0	-1
0	0	0
1	0	1
2	0	2
3	0	2
4	0	2
5	0	2
6	0	2

Absolute Value Component

The digit serial absolute value component is defined in Figure 8.

$$x_{in} \rightarrow \boxed{\textbf{ABS}} \rightarrow z_{out}$$

$$\text{where } z_{out} = \mid x_{in} \mid$$

Figure 8. MSD Digit Serial Absolute Value

The functioning of the absolute value component depends upon the sign of the most significant non-zero digit. Thus, the absolute value computation is an operation which requires its inputs most significant digit first. If the most significant digit is positive, no further arithmetic processing on the operand digits are required. If the most significant digit is negative, both that digit and all subsequent (less significant digits) will need to be negated (two's complemented). The three state, finite state machine shown in Figure 9 more completely defines the **ABS** component. In the diagram, the next output digit value is shown in parenthesis below the state transition condition. The wave control signal, W_{in} , which is active on arrival of the most significant digit of the operands, is used to initialize the finite state machine to state A.

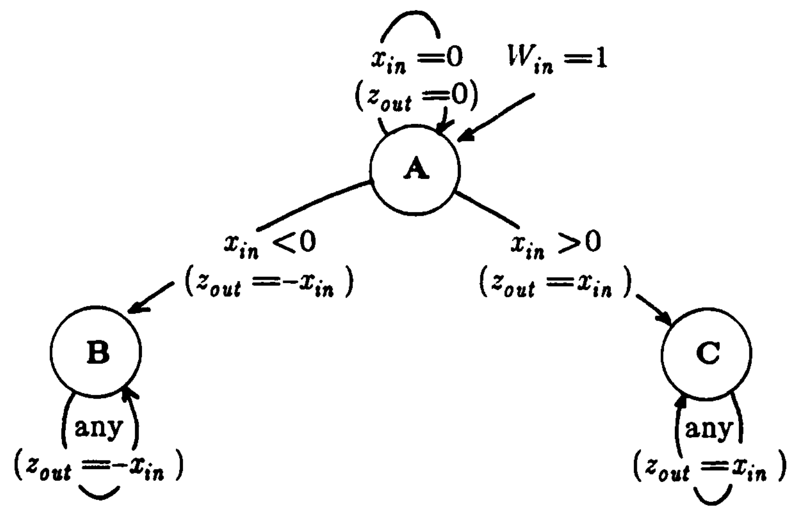


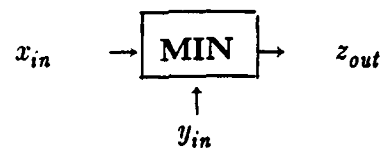
Figure 9. Finite State Machine for **ABS**

For example, when $X_{in} = \bar{1}232_4$ ($= -0102_4$) the absolute value component would output $Z_{out} = 1\overline{232}_4$ ($= 0102_4$). Although the result digit can be determined based on only the current input digit, the latency of the **ABS** component is defined to be one to be consistent with the other primitive components. Thus, the result digit is held until the next clock cycle before it is output. The **ABS** primitive component has

been estimated at 150λ by 150λ in custom logic, double metal CMOS. As with **ADD**, only one component is needed regardless of the operand precision. While this operation could be combined with the subtract component to build an absolute value subtractor, we discuss them here as separate components for simplicity.

Minimum Component

The digit serial minimum component which computes the minimum of two input values is defined in Figure 10.



$$\text{where } z_{out} = MIN\ (x_{in}, y_{in})$$

Figure 10. MSD Digit Serial Minimum

Finding the minimum of two operands, an operation that is very straightforward when dealing with a conventional digit set, becomes more difficult when using a signed-digit set. Consider, for example, when $X_{in} = 12\bar{3}2_4$ and $Y_{in} = 112\bar{1}_4$. One's first guess might be to choose Y_{in} as the minimum, but in fact X_{in} is the actual minimum value. Recall that in digit serial processing with a latency of one, the hardware can only "look ahead" one digit position. The five state, finite state machine [Nas] given in Figure 11 more completely defines the **MIN** component. The wave control signal W_{in} is once again used to initialize the finite state machine to state **A**.

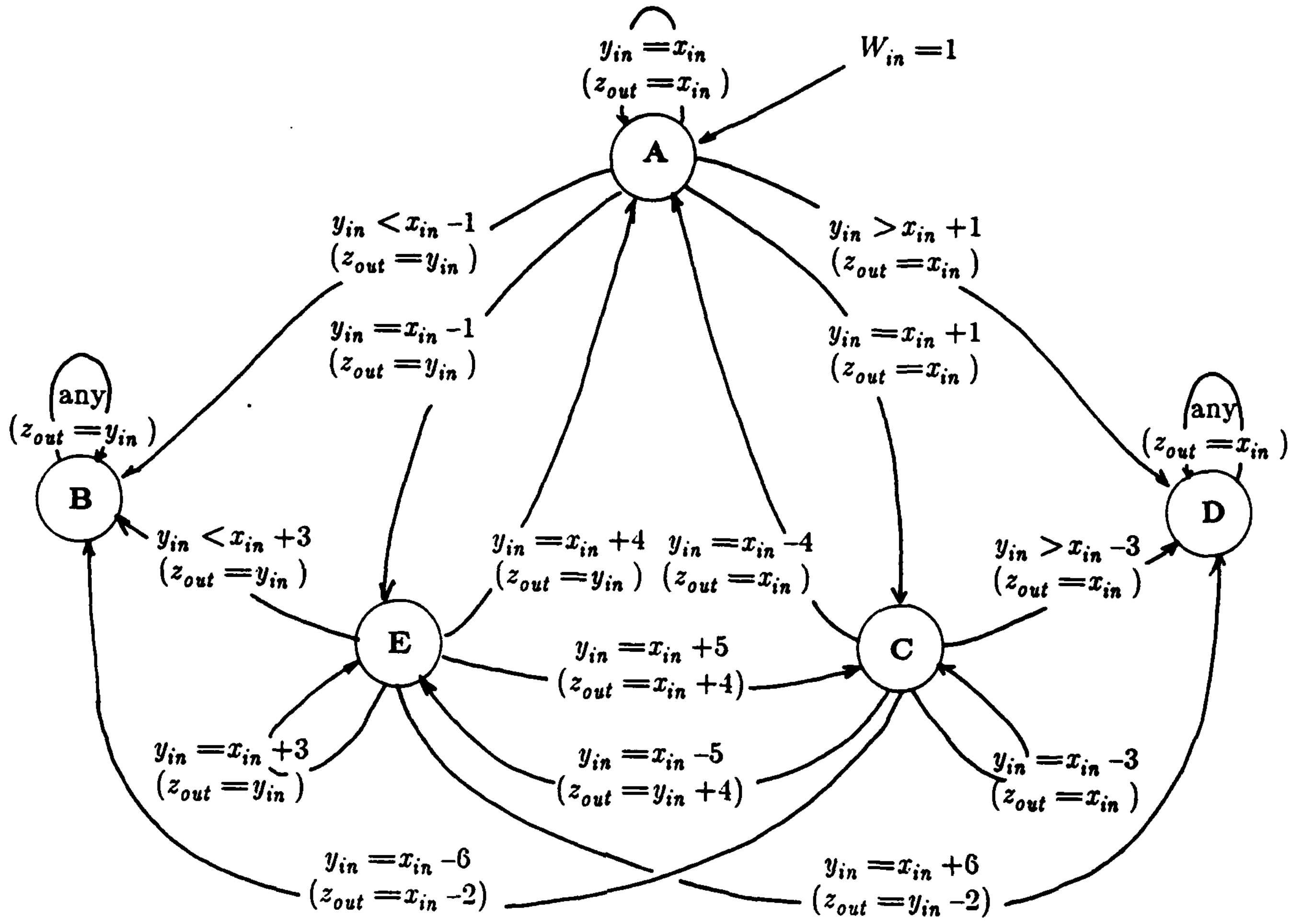


Figure 11. Finite State Machine for **MIN**

As with the **ABS** component, the latency of the **MIN** component is defined to be one. Logic equivalent to a one digit adder cell is needed to determine both the next state and the correct output digit. The **MIN** primitive component has been estimated at 300λ by 200λ in custom logic, double metal CMOS. As with **ADD** and **ABS**, only one component is needed regardless of the operand precision. Three **MIN** components are needed to find the minimum of three values as required in the DTW PE.

Using the finite state machine in Figure 11 two msd, base 4, digit serial minimum examples are illustrated in Table 4. The first **MIN** operation where $X_{in} = 12\bar{3}2_4$ and $Y_{in} = 112\bar{1}_4$ outputs $Z_{out} = 1112_4$, a numerically equivalent, recoded version of X_{in}, the actual minimum input value. The second **MIN** operation where $X_{in} = \bar{1}232_4$ ($= -0102_4$) and $Y_{in} = \bar{1}222_4$ ($= -0112_4$) correctly outputs the numerically minimum value $Z_{out} = \bar{1}222_4$.

Table 4. Digit Serial Minimum Example

clock	1	2	3	4	5	6	7	8	9
W_{in}	1	0	0	0	1	0	0	0	*
x_{in}	1	2	-3	2	-1	2	3	2	*
y_{in}	1	1	2	-1	-1	2	2	2	*
present state	a	a	e	c	a	a	a	e	*
z_{out}	*	1	1	1	2	-1	2	2	2
next state	a	e	c	c	a	a	e	b	*

Digit Pipelined DTW Processing Element

Our dynamic time warp PE is built out of the primitive arithmetic components described in the previous section. Recall that each PE in the DTW processor must be able to compute

$$S_{i,j} = \mid U_i - R_j \mid + \text{minimum}(S_{i-1,j}, S_{i-1,j-1}, S_{i,j-1})$$

$$= \textbf{ADD}(\textbf{ABS}(\textbf{SUB}(u_i, r_j)) + \textbf{MIN}(\textbf{MIN}(s_{i-1,j}, s_{i-1,j-1})\textbf{MIN}(s_{i-1,j-1}, s_{i,j-1}))) .$$

The internal block diagram description of one such PE is shown in Figure 12.

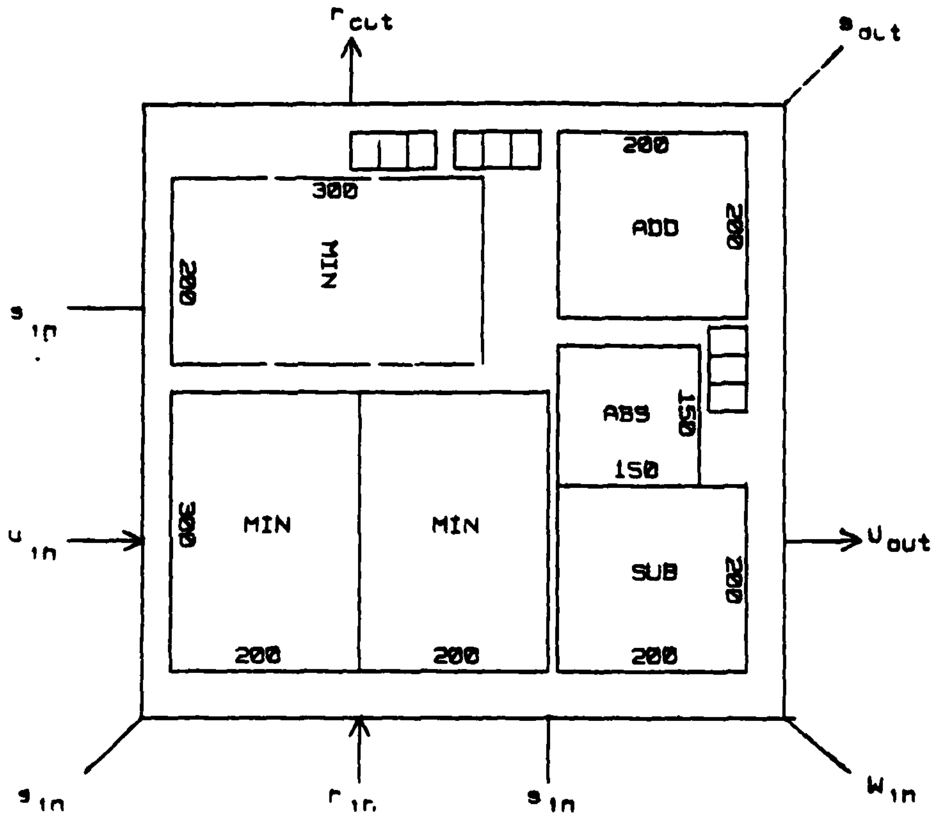


Figure 12. Block Diagram of the DTW PE

Each PE contains, in addition to three **MIN** components, one **ADD** component, one **SUB** component, and one **ABS** component, three three-bit registers for latching the digit values of r_{out}, u_{out}, and s_{out}. A wave control signal, W_{in}, is broadcast to all of the PE's on the same back diagonal and, therefore, is not latched through the PE. Using the estimates for the sizes of the primitive arithmetic components given in the previous section and a $\lambda = 1.5$ microns for 3 micron feature size devices, the size of the PE is estimated at under 1000 microns by 1000 microns.

The clocking scheme proposed for the PE is shown in Figure 13. Three two-phase clock pulses are required to completely process one digit through the PE. **MIN**($s_{i-1,j}$, $s_{i-1,j-1}$), **MIN**($s_{i-1,j-1}$, $s_{i,j-1}$) and **SUB**(u_i, r_j) are done during the first cycle, **MIN**(**MIN**($s_{i-1,j}$, $s_{i-1,j-1}$), **MIN**($s_{i-1,j-1}$, $s_{i,j-1}$)) and **ABS**(**SUB**(u_i, r_j)) are done during the second cycle, and the final addition is done during the third cycle. The wave pulse, W_k, will reoccur with the arrival of the most significant digit of the next coefficient or every 24 clock pulses for eight digit (16-bit) operands.

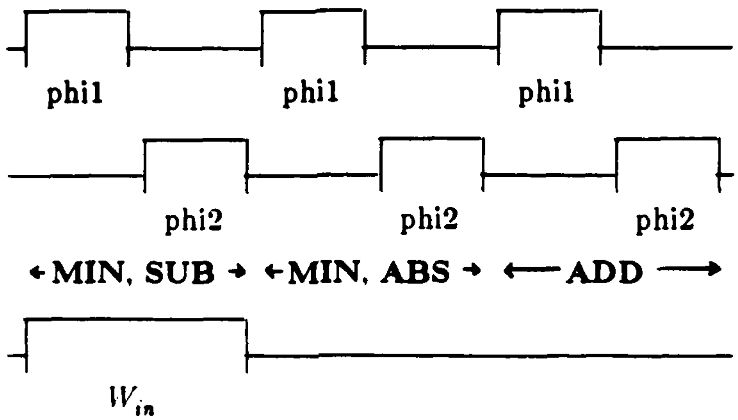


Figure 13. PE Clocking Scheme

Assuming a gate delay of two nanosecond, each clock phase should be active for eight nanoseconds since each primitive arithmetic component cell contains four levels of logic. Referring, for example, to the logic in the **ADD** component shown in Figure 6, digit operation (1) would be performed during phi1 and digit operation (2) during phi2 of the appropriate clock cycles. Allowing one nanosecond phase isolation, the PE would require 54 nanoseconds to process one digit which we refer to as the PE cycle time. Since the primitive arithmetic components are all digit pipelined, the next digit of the operands are processed by this PE during the next PE cycle while the results of this PE are passed to its neighbors for processing.

Digit Pipelined DTW Processor

Obviously, the entire 504 PE DTW processor shown in Figure 4 will not fit on one VLSI chip in today's technology. A partitioning of the processor into several chips, considering both chip area and pin limitations, must be done. Available MOSIS standard frames [MOS] for a single VLSI chip which are viable candidates for the DTW processor are shown in Table 4.

Table 4. Available MOSIS Frames (3 micron feature size)

Technology	Max. Project Size (microns)	No. of Pins (grid array)
CMOS/Bulk	4600 x 6800	84
CMOS/Bulk	6900 x 6800	84
CMOS/Bulk	7900 x 9200	84

Various partitionings of the DTW into chips were considered. The most promising one, shown in Figure 14, contains three chips types all of which fit into MOSIS stan-

dard frames. Chip type A contains a complete three by three array of PE's. Chip types B and C, which are only partially populated with PE's, are mirror images of one another.

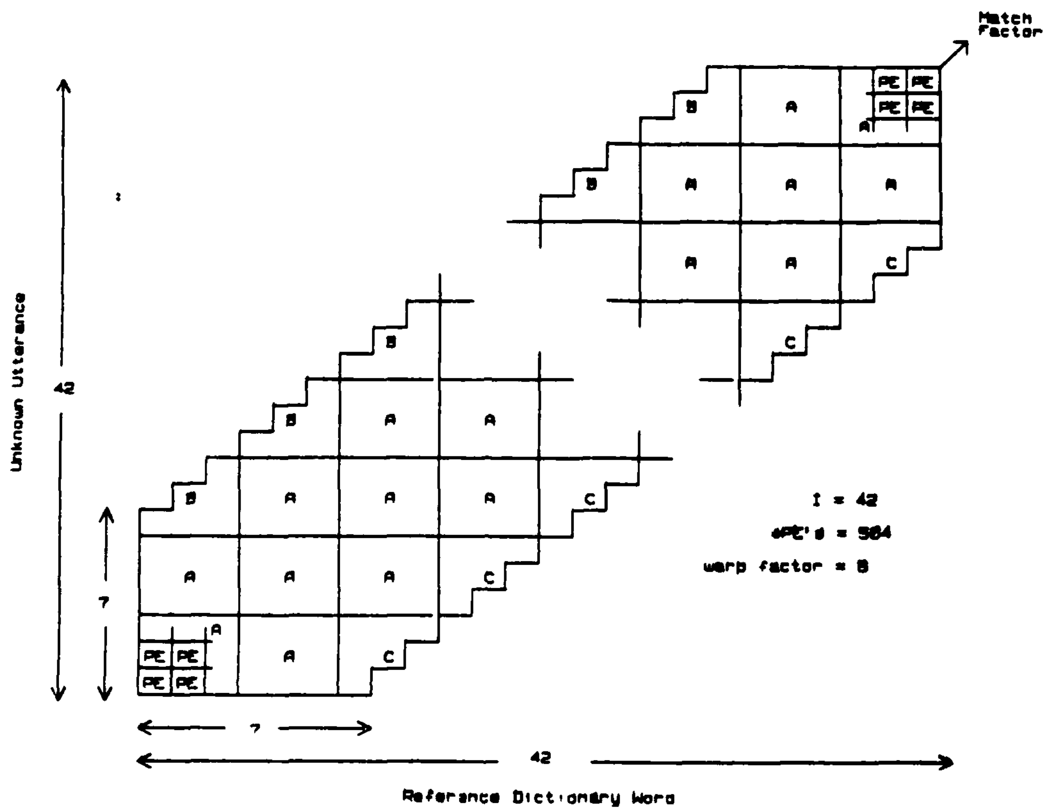


Figure 14. DTW Partitioning (I=42)

Using this partitioning, 40 chips of type A, 12 of type B, and 12 of type C for a total of 64 chips are required to implement the full DTW processor of I = 42 and a warp factor of six. Table 5 gives a sampling of the number of chips which would be required for various frame counts and warp factors. For example, the entire 1764 PE mesh would be needed to process connected speech [QGG].

Table 5. Chip Counts						
I	warp factor	#PEs	#chips type A	#chips type B	#chips type C	TOTAL #chips
42	21	1764	196	0	0	196
42	6	504	40	12	12	64
33	6	387	31	9	9	49
24	6	270	22	6	6	34
24	3	156	8	7	7	22

Not only are the PE's nearest neighbor interconnected within each of the chip types, but the chips are also interconnected in a nearest neighbor fashion at the board or wafer level. The chip counts do not include the memory chips required to hold the reference dictionary or the control chip required to generate the broadcast wave control and clocking signals. For 42 frames, 84 wave control signals are required, one for each back diagonal (with a maximum of 10 active at one time for eight digit operands).

The block diagram of chip type A is shown in Figure 15 with data and wave control signals indicated. Globally routed reset, power, ground, and clock lines are not shown.

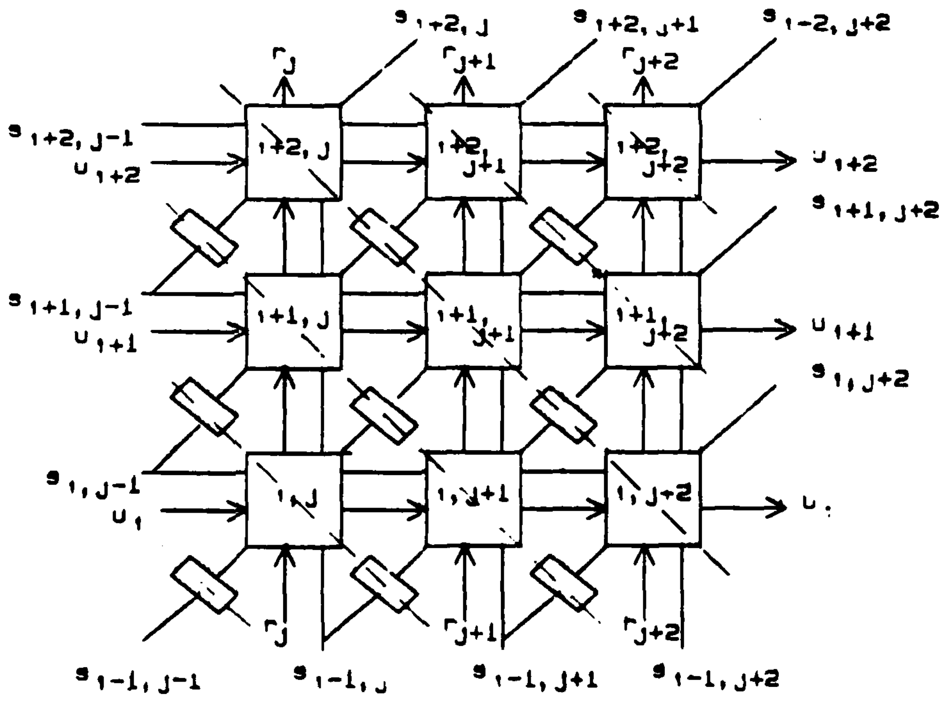


Figure 15. Block Diagram of Chip Type A

Using the area requirement of under 1000 microns by 1000 microns for each PE, MOSIS standard frame 6900 microns by 6800 microns will easily accommodate the three by three grid of PE's while leaving ample room for global routing and pads. Nine digit registers are needed to latch the data between adjacent waves of operation.

While the area requirement is easily achieved, the pin limitation is not. A count of the data requirements for chip type A is six pins for inputting r_j, r_{j+1}, and r_{j+2}, six for outputting r_j, r_{j+1}, and r_{j+2}, six for inputting u_i, u_{i+1}, and u_{i+2}, six for outputting u_i, u_{i+1}, and u_{i+2}, twenty-one for inputting the seven s_{in}'s and fifteen for outputting the five s_{out}'s for a total of 60 data lines. In addition, six wave control signals are required as well as a global reset signal, two clock lines, power, ground and substrate for a total of 12 control lines. Thus, chip type A uses 72 of the 84 available pins.

Isolated Word Recognition Rate

The recognition rate of the DTW processor can now be determined. The PE cycle time, the time for a PE to process one digit of a coefficient, is 54 nsec. With 84 waves in a 42 frame mesh, 4.536 usec are required to produce one digit of a match coefficient. This is also the amount of time which must be allowed for mesh fill time. With eight digits per coefficients, 36.288 usec are required to process each coefficient. If there are eight coefficients per frame, 290.5 usec of processing time once the mesh is full are required. Since the DTW is pipelined at the digit level, once the mesh is full and the most significant digit of the first match factor coefficient is output by the DTW processor, a digit is output every 54 nsec. Assuming that a real-time recognition rate requires that all of the match factors for the entire reference dictionary can be generated in about half a second, our DTW processor can process a reference dictionary of 2000 words with eight, eight digit coefficients per frame in real-time. Table 6 summarizes the word recognition rate of the 42 frame DTW processor for various numbers of coefficients per frame of various precisions. The only impact on the DTW hardware when changing the number of coefficients and/or their precision, is assuring that the control chip coordinates the broadcast wave signal with the delivery of the most significant digits of the unknown utterance and the reference words to the processing element. The DTW processor design does not change.

Table 6. Word Recognition Rates for 2000 Words (I=42)		
# Coef./Frame	# Digits/Coef.	Recogn. Rate (sec)
4	3	.109
4	8	.290
6	8	.435
8	8	.581
12	8	.871
16	8	1.161
16	12	1.742

Conclusions

A custom VLSI design of a DTW processor for isolated speech recognition has been presented which is real-time for dictionaries containing 2000 words. Our design differs from previous DTW processors in that (1) the data is represented in signed-digit, base 4 format; (2) the digits are passed between processing elements in a most significant digit first, digit serial fashion; and (3) the algorithms are pipelined at the digit level with each primitive component having a latency of one. This allows a very high degree of concurrency and a high data rate to be maintained, while the pin out requirements are kept low. VLSI implementations for each of the primitive components needed to build a processing element were presented. A partitioning into chips of a 42 frame DTW processor with a warp factor of six was proposed. Our VLSI DTW processor is both flexible and modular. The design is independent of the number of coefficients per frame and the precision of those coefficients. The design is also easily expandable in the number of frames per utterance and the warp factor.

The **ADD** primitive component has been fabricated by MOSIS, while the other primitive components are in design. Future designs will use a 1.2 micron feature size now supported by MOSIS. This will reduce the area of each processing element, but still no more than nine processing elements will fit on one chip due to the pin constraints. Efforts are also underway to combine the **SUB** and **ABS** into one primitive component, as well as the three **MIN** components. This will allow us to reduce the PE cycle time to two clock cycles (32 nsec.). Primitive components to provide other local distance measures are also being investigated. The VLSI design effort is supported by a number of CAD tools under development at Penn State [ref]. Ultimately, these tools will provide automatic layout of architectures like the DTW processor from a high-level algorithmic description of the architecture.

Acknowledgements

This work has been supported in part by the Army Research Office under Contract DAAG29-83-K-0126. The cell design and layout for the **ADD** primitive component was done by Robert M. Owens. The author also wishes to thank her colleagues at the Supercomputing Research Center for their encouragement and help during her sabbatical leave when much of this paper was written.

References

Ack Ackland, B., "Dynamic Time Warp Processor," pp. 384-407, *Principles of CMOS VLSI Design,* by Weste, N. and Eshraghian, Addison-Wesley, 1985

AnB Anantharaman, R. and R. Bisiani, "A Hardware Accelerator for Speech Recognition Algorithms," pp. 216-223, *Proceeding of the 13th International Conference on Computer Architecture,* Tokyo, June 1986.

Atk Atkins, D., "An Introduction to the Role of Redundancy in Computer Arithmetic," Vol. 8, No. 6, pp. 74-76 *Computer,* June 1975.

Bee Beekman, J., "Mesh Arrays for CMOS VLSI Design," CS-86-25, Penn State University, August 1986.

Bel Bellman, R.E., *Dynamic Programming,* Princeton University Press, 1957.

BAW Burr, D., Ackland, B. and N. Weste, "Array Configurations for Dynamic Time Warping," *IEEE Transactions on ASSP,* Vol ASSP-32, No. 1, pp. 119-128, Feb. 1984.

IrO Irwin, M.J. and R.M. Owens, "Digit Pipelined Arithmetic: A Tutorial," submitted to *IEEE Computer,* April 1986.

KNL Kavaler, R., et.al., "A Dynamic Time Warp IC for a One Thousand Word Recognition System," *Proceedings of ICASSP '84,* San Diego, CA March 1984.

MaR Mann, J.R. and R.M. Rhodes, "A Wafer Scale DTW Multiprocessor," *Proceeding of ICASSP '86,* Tokyo, pp. 1557-1560, April 1986.

MOS MOSIS Crew, *MOSIS User's Manual,* USC Information Sciences Institute, 1985.

MRR Myers, C.S., L.R. Rabiner and A. Rosenber, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition," *IEEE Transactions on ASSP,* Vol. ASSP-28, pp. 623-635, Dec. 1980.

MyR Myers, C.S. and L.R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Transactions on ASSP,* Vol. ASSP-29, No. 3, pp. 284-296, April 1981.

Nas Nasri, K., "VLSI Implementation of MIN and MAX Components for a Digit Pipelined Architecture," M.S. Paper, Department of Computer Science, Penn State

University, December 1985.

OwI Owens, R.M. and M.J. Irwin, "A System for Designing, Simulating, and Testing High Performance VLSI Signal Processors," Vol. CAD-5, No. 3., pp. 240-248, *IEEE Transactions on Computer Aided Design,* (July 1986).

Owe Owens, R.E, "A VLSI Dynamic Time Warp Processor for Connected and Isolated Word Speech Recognition," *Proceedings of ICASSP '85,* March 1985.

QGG Quenot, G. et.al., "A Dynamic Time Warp VLSI Processor for Continuous Speech Recognition," *Proceedings of ICASSP '86,* Tokyo, pp. 1549-1552, April 1986.

RaS Rabiner, L.R. and R. Schafer, *Digital Processing of Speech Signals,* Prentice-Hall, 1978.

Sak Sakoe, H., "Two Level DP Matching-A Dynamic Programming Algorithm for Connected Word Recognition, *IEEE Transactions on ASSP,* Vol. ASSP-27, No. 6, pp. 588-595, Dec. 1979.

SaC Sakoe, H. and S. Chiba, "A Dynamic Programming Approach to Continuous Speech Recognition," *Proceedings of the 7th Int. Conf. Acoustics,* Budapest, Hungary, pp. 65-68, Aug. 1971.

WBA Weste, N., D. Burr, and B. Ackland, "Dynamic Time Warp Pattern Matching Using an Integrated Multiprocessing Array," *IEEE Transactions on Computers,* C-32, 8, pp. 731-744, August 1983.

YoJ Yoder, M. and L. Jamieson, "Simulation of a Word Recognition System on Two Parallel Architectures," pp. 171-179, *Proceeding of the International Conference on Parallel Processing,* August 1985.

END

10-87

DTIC